

Application No. 09/541,631 (Balkany) GAU 2172

Page 7 of 12

REMARKS—General

After this amendment, claims 1 to 17 will be pending. Claims 1-10 have been amended, new claims 11-17 have been added.

Summary of changes to claims

Claim 1: Section (b)(3) was amended to focus on the essence of the tuple-storage method: the use of a length to represent a tuple sequence.

Claim 6: Sections (a) and (b) were amended to make it clearer that leaf nodes are distinct from dictionaries, and use indexes into dictionaries instead of storing actual values, and to make it clearer that a single set of dictionaries is created, which can be shared by a plurality of trees.

Claims 5 and 10 were amended to replace closed language (i.e. "consisting of") with open language. All claims had other, non-substantive rewordings intended to improve readability.

Dependent claims 11 and 12 were added to cover more-specific ways to use the sequence length to represent a tuple sequence. Claim 11 uses a length+tuple method, and claim 12 uses the mutually-consecutive tuple method.

Claims 13, 14, and 15 were added to describe particular ways a set of dictionary values can be represented. Claim 13 uses an array of value counts to represent the counts of values, at the corresponding indexes in the dictionary. Claim 14 uses a bit array to index dictionary values. Claim 15 describes representing a subset of dictionary values by the subset of their indexes.

Claims 16 and 17 use the interior-node-size estimate described in Alternate Embodiments.

Support for claims

These claims contain no new matter.

Support for claim 1 can be found at pages 7-8, DESCRIPTION—Preferred Embodiment, 1. Tree structure. Also (65. Figure 3) Also pages 12-14, DESCRIPTION—Alternative Embodiments, Operation, 3. Record addition, Interior Node Storage, and otherwise throughout the specification.

Application No. 09/541,631 (Balkany) GAU 2172

Page 8 of 12

Support for claim 6 can be found at:

page 7, DESCRIPTION—Preferred Embodiment, 1. Tree structure (up to Definition), and at page 8, 2. Separation of dictionaries and leaves. Also (60, 61. Figure 3). Also page 11, DESCRIPTION—Alternative Embodiments, (b). Also page 13: Operation, 3. Record addition, (b), and otherwise throughout the specification.

Summary of amendments to specification

The sentence inserted into page 8 was intended to make it clearer that a run length was used to describe the run.

Other corrections to the specification fixed typographical errors.

Summary of amendments to the drawings

Two typographical errors were corrected:

Figure 10: A bracket was moved to give: "[(1, 1) (2, 2) (3, 3) (4, 4) (5, 5) (6, 6)] + (4, 4)"

Figure 16: Changed "Design (State s)" to "Initial (State s)" in the step at the top of the flowchart.

Copies of these figures before and after the corrections are attached.

The foregoing amendments are taken in the interest of expediting prosecution and there is no intention of surrendering any range of equivalents to which Applicant would otherwise be entitled in view of the prior art.

By amending the application, the Applicant does not concede that the patent coverage available to them would not extend as far as the original claim. Rather, Applicant reserves the right to file a continuation application to pursue the breadth of the claims as filed. Applicant believes that the Examiner has not made a sufficient showing of inherency of the teachings of the asserted prior art, especially given the lack of teachings in the cited references of the properties that Applicant has recited in his claims.

Further, by the present amendment, it does not follow that the amended claims have become so perfect in their description that no one could devise an equivalent. After amendment, as before, limitations in the ability to describe the present invention in language in the patent claims naturally

Application No. 09/541,631 (Balkany) GAU 2172

Page 9 of 12

prevent the Applicant from capturing every nuance of the invention or describing with complete precision the range of its novelty or every possible equivalent. See, *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.*, 62 USPQ2d 1705 (2002). Accordingly, the foregoing amendments are made specifically in the interest of expediting prosecution and there is no intention of surrendering any range of equivalents to which Applicants would otherwise be entitled.

Rejection of claims 1-10 under 35 U.S.C. 102(b)

Claims 1-10 were rejected as being anticipated by Bugajski (US. Patent No. 5,592,667). The Applicant response respectfully asserts that the Office Action is incorrect.

1. Regarding on claim 1.

TWO POINTS:

a) The use of a tuple-sequence length is unique

In Description—Preferred Embodiment, Part 1, pages 7-8, the storage of mutually-consecutive tuples is described. The key idea is that the count of tuples in this run is used to represent multiple tuples with a single entry

For example, in Figure 3, item 65 in the application, a **single entry represents nine tuples**. The nine tuples require nine entries in prior art, as seen in Figure 2, item 56, the same nine tuples are represented as the nine rows, starting with Token 2. Thus, the present invention provides a substantial improvement in compression over the prior art.

Nowhere does Bugajski teach the concept of the length of a tuple sequence as a means to represent a plurality of tuples.

b) The use of mutually-consecutive tuples is unique

Mutually-consecutive tuples, which are defined at the bottom of Page 7, describe a pattern of (left, right) value pairs that is seen in interior nodes. (For example, see the sequence of value pairs in the node, (56, Figure 2). This regularity allows the sequence of multiple tuples to be described by its start element and length.

Nowhere does Bugajski teach the concept of runs of mutually-consecutive tuples.

Application No. 09/541,631 (Balkany) GAU 2172

Page 10 of 12

Compression is new and unexpected result

In the preceding example, a 9-fold compression was described. The following example demonstrates a **million-fold** compression. This degree of compression is new and unexpected.

Example

The difference between the methods can be illustrated by an example. Assume there is a run of one million mutually-consecutive tuples: (0, 0), (1, 1), (2, 2) ... (999,999, 999,999).

Bugajski's method, diagrammed in Bugajski (Figure 2), requires two integers and one bit for each tuple. This would take space for two million integers and one million bits. Assuming 4 bytes per integer and 8 bits per byte, this is 8,125,000 bytes.

The Applicant's method, diagrammed in Balkany (65, Figure 3), only requires four integers for a tuple run of *any* length. This would be only 32 bytes, a substantial improvement! The method is explained at the bottom of page 7 and the top of page 8.

With respect to claim 1, in addition to not showing the use of a sequence length to represent a plurality of tuples, Bugajski also doesn't show the use of mutually-consecutive tuples. For these reasons, Bugajski doesn't anticipate or otherwise make obvious claim 1.

Regarding on claim 3. The Office Action asserted that this claim was described in US patent 5,592,667 (Bugajski)(col. 9, lines 55-69, and col. 12, lines 55-67) . The Applicant respectfully asserts that the Office Action is incorrect.

The gate field is unique

As described in Preferred-Embodiment (3)(Balkany), each interior node has a gate field associated with it **in addition to** Bugajski's "table of associative memories".

Search efficiency is new and unexpected result

Often a subset of a tree's leaves must be processed for each operation on the data. Standard tree-traversal algorithms will visit every node, which wastes time for each leaf visited which is not

Application No. 09/541,631 (Balkany) GAU 2172

Page 11 of 12

in the subset to be processed.

A gate field that tells which branches lead to leaves in the set allows branches that don't lead to leaves to be skipped, improving search efficiency. For example, Figure 6 shows a tree with the gate field value shown for non-leaf nodes. The non-zero gate-field values indicate the nodes, which are ancestors of leaves, in {A, E, F}, (the subset of leaves to be processed). The eight nodes visited for each operation using leaves A, E, and F are shaded in the figure. The gate field allows each operation to be done visiting 53% of the tree's nodes, compared to 100% for a standard depth-first-search algorithm.

Nowhere does Bugajski teach the concept of a gate field to improve search efficiency. His figures 2, 3, and 4 show interior nodes consisting solely of associative memories, with no gate field. In addition to reasons discussed with respect to claim 1, Bugajski also doesn't anticipate or otherwise make obvious claim 3.

2. Regarding on claim 6. The Office Action asserts that Bugajski (col. 9, lines 56-61) teaches Claim 6. The Applicant respectfully asserts that the Office Action is incorrect.

This invention uses leaf nodes with only indexes, not values

Claim 6, Step (b)(1) recites that "said leaf nodes are distinct from said dictionaries and each said leaf node represents is capable of representing a subset values from one of said dictionaries using numeric indexes into said dictionary ...".

In (61, Figure 3), these items are leaves using **only the indexes** of values stored in separate dictionaries (60, Figure 3). Each leaf-node entry indexes the dictionary value at that position number.

Bugajski doesn't teach the concept of leaf nodes with only indexes. As Figure 3 (Bugajski) shows, Bugajski's leaf nodes also contain the actual field values corresponding to the indexes. (The two leftmost nodes in the figure are leaves, labeled B and A.)

A row of each is shown, giving the field value in addition to the index. The leaves here are the dictionaries. Thus Bugajski does not show leaves that are separate from dictionaries.

Application No. 09/541,631 (Balkany) GAU 2172

Page 12 of 12

Separation of leaves and dictionaries improves compression for plurality of trees

An Index (whether a token, count, or bit) is generally much shorter than a field value. For example, see Figure 3 of the present application. The contents of the leaves (61, Figure 3) are just counts for the dictionary values at the same indexes, which are a fraction of the size of the dictionaries (60, Figure 3).

Often it's necessary to manipulate multiple trees from the same data set, as when extracting selected records from one tree and inserting them into a second one. Separation of leaves and dictionaries allows the leaves from a plurality of trees to share one set of common dictionaries.

This avoids having to duplicate each dictionary for each tree, which wastes space, since the dictionaries are all from the same data set. Since the dictionaries generally require more space than the indexes, this saves a considerable amount of space, improving compression when a plurality of trees are processed simultaneously.

In addition, space may also be saved when one of the trees sharing a dictionary set only contains a small subset of the values.

Nowhere does Bugajski teach the concept of separation of leaves and dictionaries. For these reasons, Bugajski doesn't anticipate or otherwise make obvious claim 6.

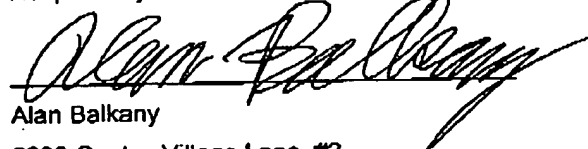
Conclusion

The Applicant's claims demonstrate unique material and are patentable. Please reconsider this application.

2004 JAN 29

Date

Respectfully submitted,



Alan Balkany

5693 Cooley Village Lane, #2

Waterford, MI 48327

(248) 681-6268

alankdkd@aol.com